**Логика проекта**

Проект «FinanceManager» предназначен для управления личными финансами пользователей. Он позволяет пользователям отслеживать свои доходы и расходы, устанавливать месячный бюджет и анализировать финансовую историю. Приложение предоставляет графический интерфейс, который упрощает взаимодействие пользователя с функциональностью программы.

**Основные функции приложения:**

* **Добавление доходов:** Пользователь может вводить различные источники дохода, такие как зарплата, премии и подарки.
* **Добавление расходов:** Пользователь может регистрировать различные категории расходов, такие как еда, транспорт и развлечения.
* **Установка бюджета:** Пользователь может установить месячный бюджет и отслеживать текущие расходы в рамках этого бюджета.
* **История операций:** Приложение ведет учет всех финансовых операций, позволяя пользователю просматривать и редактировать их при необходимости.
* **Уведомления о превышении баланса**: Программа предупреждает пользователя о превышении текущего баланса.

**Структура проекта**

Проект состоит из нескольких основных модулей:

**1. Модуль интерфейса (UI).**

**Файл UI:** finance\_manager.ui, созданный с помощью Qt Designer, содержит графический интерфейс приложения.

**Класс FinanceManager:** Основной класс приложения, который наследует от QMainWindow и загружает UI файл.

**2. Основные компоненты класса FinanceManager.**

Инициализация переменных:

**self.balance:** Текущий баланс пользователя.

**self.income\_history:** Список всех доходов.

**self.expense\_history:** Список всех расходов.

**self.monthly\_budget:** Установленный месячный бюджет.

**self.current\_month\_expenses:** Общая сумма расходов за текущий месяц.

Методы класса:

**\_\_init\_\_():** Конструктор класса, загружает UI и инициализирует переменные.

**add\_income():** Метод для добавления дохода в историю.

**add\_expense():** Метод для добавления расхода в историю.

**set\_monthly\_budget():** Метод для установки месячного бюджета.

**delete\_history\_entry():** Метод для удаления записи из истории операций.

**edit\_history\_entry():** Метод для редактирования существующей записи в истории.

**update\_balance():** Метод для обновления текущего баланса на основе истории доходов и расходов.

**update\_history():** Метод для обновления текстовой истории операций в интерфейсе.

**3. Взаимодействие между компонентами.**

Элементы интерфейса (кнопки, поля ввода) подключены к методам класса через сигналы и слоты PyQt5. Например:

* Кнопка «Добавить доход» вызывает метод **add\_income()**.
* Кнопка «Добавить расход» вызывает метод **add\_expense()**.
* Все изменения в данных (баланс, история) автоматически обновляют соответствующие элементы интерфейса (например, текстовые метки и текстовые поля).